

Graph based Version of KNN in Combination of Text Segmentation with Text Categorization

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the graph based KNN version for automating the text segmentation by combining it with the text categorization. In the previous work, we adopted the graph-based version as an approach to the text segmentation, but must present the domain as a tag, manually. In this research, we also adopt the KNN algorithm which receives a graph as the input data, as the approach to the text segmentation and the text classification. In the proposed system, only a text without its domain is given as the input, it is classified into a domain by the text classification, and each paragraph pair is classified into boundary or continuance by the classifier which corresponds to the domain. This research is intended to automate completely the text segmentation by introducing the text classification, and to improve the performance of both tasks, using the graph based KNN algorithm.

I. INTRODUCTION

The text segmentation which is covered in this research is defined as the process of segmenting a text into subtexts by the topic transition degree. Because each boundary is established between two paragraphs, the text segmentation is viewed into a binary classification of adjacent paragraph pairs into boundary or continuance. The process of text segmentation is to partition a text into paragraphs, take adjacent paragraph pairs by sliding a two-sized window on paragraphs, and classify adjacent paragraph pairs into boundary or continuance. Boundaries are put between paragraph pairs which are classified into boundary, and the text is segmented following the boundaries. The classification which is mapped from the text segmentation is a domain dependent task where a same entity is classified differently depending on the domain, so it is necessary to present the domain as the input together with the input text.

This research is motivated by the requirement of presenting a text with its own domain for segmenting a text. The text segmentation is viewed into independent classification tasks, each of which corresponds to its own domain. We need to know the domain in advance for doing the text segmentation. It is possible to present only a text without its domain by automating the process of assigning a domain to a text through the text classification. The problem is solved by connecting the text segmentation with the text classification.

In this research, the text segmentation relates to the text classification, and the graph based KNN algorithm is adopted as the approach to both tasks. We define the similarity metric between graphs and use it for modifying the KNN algorithm into the graph-based version. It is applied to the text classification whose role is to decide a domain automatically to an input text. Two adjacent paragraphs are classified into continuance or boundary by the graph-based version. In this research, we propose the graph based KNN algorithm for implementing the text segmentation and the text classification.

Let us mention some benefits from this research. Encoding text into graphs is the solution to the problems in encoding them into numerical vectors. The performance of the text classification and the text segmentation is improved by adopting the graph based KNN algorithm. The process of assigning a domain to a text for activating the text segmentation. We provide the chance of modifying other machine learning algorithms into graph-based versions.

Let us mention remaining tasks as further discussions on this research. We may define and characterize mathematically other operations on graphs. Paragraph pairs and texts are encoded into compound representations, each of which consists of more than one structured form. Other machine learning algorithms and deep learning algorithms may be modified into graph-based versions. The text segmentation system which relates to the text classification may be implemented as a real program or a library by adopting the graph based KNN algorithm.

II. PROPOSED WORKS

A. Encoding Proccerss

This section is concerned with the graph as a text representation and the similarity between graphs. In representing a text into a graph, a word is given as a vertex, and a similarity between words is given as an edge. A graph is viewed as a set of edges, and the similarity between edges is computed before computing the similarity between graphs. The similarity between them is computed by averaging the similarities of all possible edge pairs. This section is intended to describe the process of encoding a text into a

graph and the process of computing the similarity between graphs.

The process of encoding a text into a graph is illustrated in Figure 1. Words are retrieved as the vertices from the text by indexing it. The similarities among words are computed as edges in the edge definition. In the edge selection, the edges with their higher similarities are selected for representing a text into a graph. Refer to [1] for studying the encoding process in more detail.

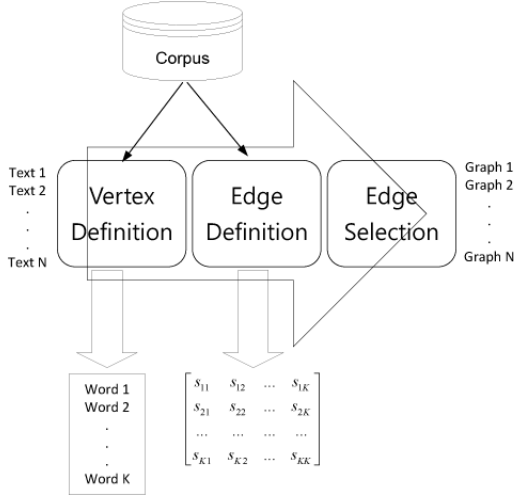


Figure 1. Process of Encoding Texts into Graphs

The three cases of computing the similarity between two edges are illustrated in Figure 2. Each weight which is associated with its own edge is assumed to be a normalized value between zero and one, and if both nodes are identical to each other, the average over two weights is the similarity between two edges. If either of the two nodes is identical to each other, the product of two weights is the similarity between two edges. If neither of two nodes is identical, zero is the similarity between them. The similarity between two edges is always a normalized value between zero and one.

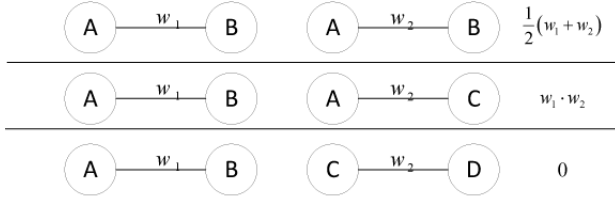


Figure 2. Three Cases of Comparing Two Edges with Each Other

Let us mention the process of computing the similarity between graphs as a semantic similarity between texts. The two graphs which represent texts are notated by edge sets, $G_1 = \{e_{11}, e_{12}, \dots, e_{1|G_1|}\}$ and $G_2 = \{e_{21}, e_{22}, \dots, e_{2|G_2|}\}$. Two edges, $e_{1i} \in G_1$ and $e_{2j} \in G_2$, are associated with their own weights, w_{1i} and w_{2j} , and the similarity between two

edges, e_{1i} and e_{2j} by equation (1), (2), or (3), depending on the cases which are mentioned above,

$$\text{sim}(e_{1i}, e_{2j}) = \frac{1}{2}(w_{1i} + w_{2j}) \quad (1)$$

$$\text{sim}(e_{1i}, e_{2j}) = w_{1i} \times w_{2j} \quad (2)$$

$$\text{sim}(e_{1i}, e_{2j}) = 0 \quad (3)$$

The similarity between two graphs is computed by equation (4),

$$\text{sim}(G_1, G_2) = \frac{1}{|G_1| \times |G_2|} \sum_{i=1}^{|G_1|} \sum_{j=1}^{|G_2|} \text{sim}(e_{1i}, e_{2j}). \quad (4)$$

The value which is generated from equation (4), is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(G_1, G_2) \leq 1. \quad (5)$$

Let us make some remarks on the encoding process and the computation of the similarity between texts. A text is encoded into a graph by the vertex definition, the edge definition, and the edge weighting. The three cases are considered for computing the similarity between edges. The similarity between graphs is computed by equation (4). In the future research, we consider representing a text into multiple graphs.

B. Graph based KNN Algorithm

This section is concerned with the graph based KNN algorithm. It was initially proposed as the approach to the word classification by Jo in 2018 [2]. The similarity metric between graphs which was described in the previous section is used for computing the similarity between a novice graph and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 3. The labeled words which are gathered as samples are encoded into graphs, and they are notated by a set, $Tr = \{G_1, G_2, \dots, G_N\}$. A novice word is encoded into a graph notated by G . Its similarities with the graphs which represent the sample words are computed by equation (4), as $\text{sim}(G, G_1), \text{sim}(G, G_2), \dots, \text{sim}(G, G_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice

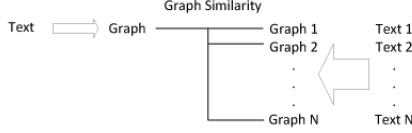


Figure 3. Similarities of Novice Input with Training Examples

example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = \text{Select}_k(Tr, G), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{G_1^{near}, G_2^{near}, \dots, G_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

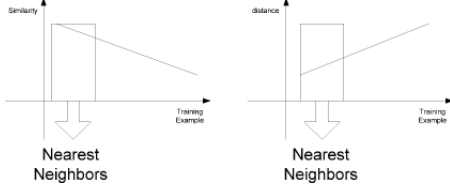


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(G_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, G , is decided by equation (8),

$$\text{label}(G) = \arg\max_{j=1}^m \text{Count}(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.

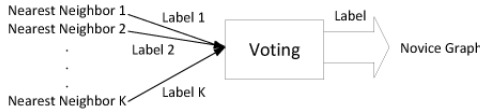


Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the graph based KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the text segmentation system which adopts the graph based KNN algorithm. In Section II-B, we described the proposed version of KNN algorithm as the approach to the text segmentation. It is viewed as a binary classification which classifies a paragraph pair into boundary or continuance. This section is intended to describe the text segmentation system with respect to its function and its architecture.

The process of gathering sample paragraph pairs and classifying a novice one, is illustrated in Figure 6. Because even a same paragraph pair may be classified differently depending on the domain, the task is called domain dependent classification. Sample paragraph pairs are gathered domain by domain, and each pair is labeled with boundary or continuance. The paragraph pairs are taken from texts which are tagged with their same domain, each paragraph pair is classified into boundary or continuance. Sample paragraph pairs which are labeled with one of the two categories are encoded into graphs in each domain.

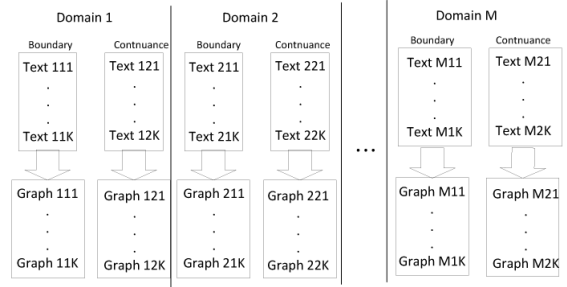


Figure 6. Preparation of Training Examples

The entire architecture of the proposed text segmentation system is illustrated in Figure 7. A text is given as the input, and adjacent paragraph pairs are extracted by partitioning the text into paragraphs and slide the two sized window on them. The sample paragraph pairs in the boundary group and the continuance group and ones which extracted from the text are mapped into graphs in the encoding module. The paragraph pairs from the text are classified into one of the two categories in the similarity computation module and the voting module. A boundary is put between paragraphs in each pair which is classified into boundary in the text, and it is partitioned into subtexts by the boundary.

The execution flow of the text segmentation system is illustrated in Figure 8. Texts with same domain are initially collected, adjacent paragraph pairs are extracted from them, and the paragraph pairs are labeled with boundary or continuance. From an input text, adjacent paragraph pairs are extracted, and are encoded into graphs, together with the sample paragraph pairs. The adjacent paragraph pairs are classified into boundary or continuance, and there are two groups of paragraph pairs: boundary group and continuance

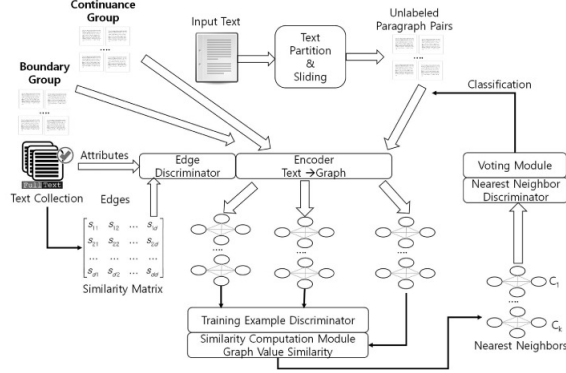


Figure 7. System Architecture

group. The boundary is marked between paragraphs in each pair in the boundary group, and text is partitioned by the marked boundaries into subtexts as the final output of this system.

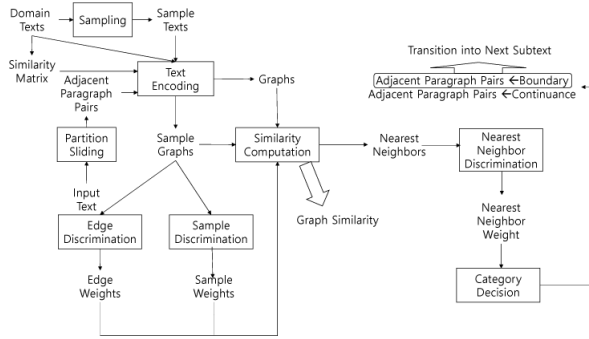


Figure 8. System Flow

Let us make some remarks on the proposed system which is illustrated in Figure 7 as its architecture. The text segmentation is defined as the binary classification where each adjacent paragraph pair is classified into boundary or continuance. The paragraph pairs are encoded into graphs instead of numerical vectors, and they are classified into directly. The input text is partitioned into subtexts by the boundary which is put between paragraphs which are classified into boundary. The subtexts as the semantic division of the input text are treated as independent texts.

D. Connection with Text Classification

This section is concerned with the connection of the text segmentation with the text classification. In the previous section, we mentioned the text segmentation system as an instance of domain dependent classification. The domain is automatically decided for nominating a classifier by connecting the keyword with the text classification. The KNN algorithm which was described in Section II-B is used for implementing the text classification. This section is intended to describe the connection of the text segmentation system

with the text classification for automating the decision of the domain.

The system architecture of the text categorization system which consists of the encoding module, the similarity computation module, and the voting module is illustrated in Figure 9. The encoding module is for encoding texts into graphs by the process which is described in Section II-A. The similarity computation module is for computing the similarities between graphs which represent a novice text and a sample text and selecting ones with their highest similarities as the nearest neighbors. The voting module is for deciding the label of the novice item by voting the labels of the nearest neighbors. This system used for automating deciding the domain of the input text.

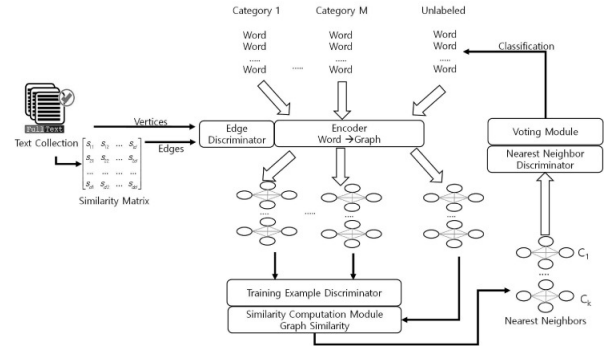


Figure 9. Text Categorization System

The connection of the text segmentation with the text classification is illustrated in Figure 10. The texts each of which is labeled with its own domain are prepared as the sample texts, and the approach to the text classification is trained with them. A domain is assigned to a novice text by classifying it into a domain automatically. The novice text is provided with its domain to the text segmentation system which is described in Section II-C. The role of the text classification system is to nominate the classifier which corresponds to the domain for the text segmentation, automatically.

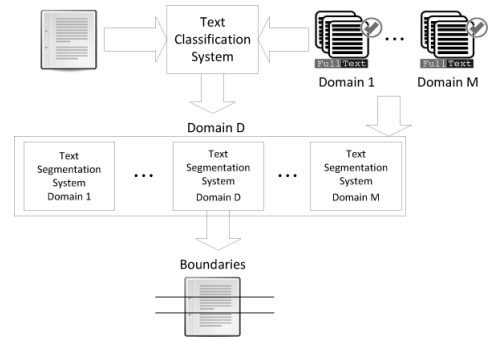


Figure 10. Text Segmentation System connected with Text Categorization

The process of executing the text segmentation, connect-

ing it with the text classification is illustrated in Figure 11. The sample texts each of which is labeled with its own domain, and sample paragraph pairs each of which is labeled with boundary and continuance are prepared in each domain. The novice text is classified into one among the predefined domains, and the classifier which correspond to the domain is nominated. Adjacent paragraph pairs are generated by sliding the two sized window on the paragraph list, and each paragraph pair is classified into boundary or continuance. Subtexts which are generated by segmenting the text following the boundaries are generated as the final output; the domain which is assigned to the input text is the guide for selecting a classifier.

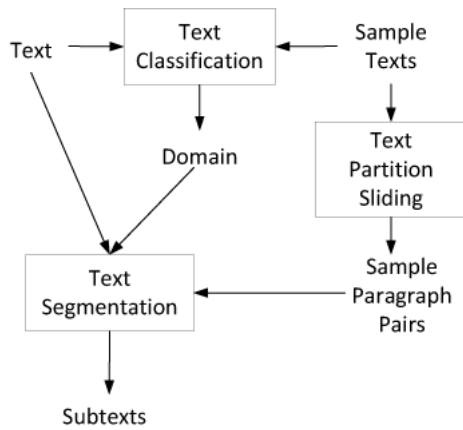


Figure 11. System Flow of Text Segmentation connected with Text Categorization

Let us make some remarks on the connection of the text segmentation with the text classification. The text classification is applied to the automatic decision of the input text domain. The role of text classification is to decide the domain for nominating a classifier, and the role of the text segmentation is to partition the entire text into subtexts based on its content. In the execution flow, the input text is classified into a domain, paragraph pairs are generated by sliding the two sized window, and each paragraph pair is classified into boundary or continuance. We consider connecting the text classification as the main task the text segmentation as the opposite to what is proposed in this research.

REFERENCES

- [1] T. Jo, "Graph based KNN for Optimizing Index of News Articles", 53-62, Journal of Multimedia Information System, 3, 2016.
- [2] T. Jo, "Comparing Graph based K Nearest Neighbor with Traditional Version in Word Categorization in NewsPage.com, 12-18, International Journal of Advanced Social Sciences, 1, 2018.
- [3] T. Jo, "Specializing K Nearest Neighbor for Content based Segmentation of News Article by Graph Similarity Metric", 9-12, The Proceedings of International Conference on Green and Human Information Technology Part II, 2019.